

In the Claims:

Please amend Claims 1 and 5 and add new Claims 9-11, as indicated below.

The status of all claims is as follows:

1. (Currently Amended) A parallel processing method for an inverse matrix for a shared memory type scalar parallel computer, comprising:

(a) specifying a square block in a matrix for which an inverse matrix is to be obtained and which is stored in a memory area as an array $A(1:n, 1:n)$ (where n is an order of the matrix) and storing the square block into a subarray $A(nbase+1:nbase+iblk, nbase+1:nbase+iblk)$ (where $nbase=iblk*(i-1)$, i is a number of repetition, and $iblk$ is a block width);

(b) decomposing the matrix into upper left, left side, lower left, upper, lower, upper right, right side, and lower right blocks surrounding the square block positioned in the center and storing each divided block into subarrays $A(1:nbase, 1:nbase)$, $A(nbase+1:nbase+iblk, 1:nbase)$, $A(nbase+iblk+1:n, 1:nbase)$, $A(1:nbase, nbase+1:nbase+iblk)$, $A(nbase+iblk+1:n, nbase+1:nbase+iblk)$, $A(1:nbase, nbase+iblk+1:n)$, $A(nbase+1:nbase+iblk, nbase+iblk+1:n)$, $A(nbase+iblk+1:n, nbase+iblk+1:n)$, respectively ;

(c) dividing the lower, right side, and lower right blocks of the square block along a longer direction of a row direction or a column direction of each block, assigning each divided block and the square block to each processor and LU decomposing each assigned block by a parallel operation of the processors;

(d) updating the left side, upper, lower, and right side blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor, the parallel operation including execution of a recursive program comprising (1) dividing blocks into a first half and a second half, (2) calculating the first half, (3) applying (1) through (3) to the second half, and further updating in parallel using the blocks updated in the recursive program on the upper left, lower left, upper right, and lower right blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor;

(e) updating the square block in plural stages using one processor by dividing the predetermined square block so that a cost of updating the predetermined square block is within about 1% against a whole cost of calculation; and

(f) setting `__nbase` of the subarray `A(nbase+1:nbase+iblk, nbase+1:nbase+iblk)` to `nbase+iblk`, and obtaining an inverse matrix of the matrix by repeating the steps (a) through (f).

2. (Original) The method according to claim 1, wherein
said shared memory type scalar parallel computer comprises a plurality of processors, plural units of cache memory provided for respective processors, plural units of shared memory, and an interconnection network for connection to be made such that the units can be communicated.

3. (Original) The method according to claim 1, wherein

said method is used to realize a Gauss-Jordan method for parallel computation for each block.

4. (Original) The method according to claim 1, wherein

a width of a division used when each block is divided for parallel computation is set such that a total amount of computation of square blocks not processed in parallel can be about 1% of entire computation from a size of a matrix from which an inverse matrix is obtained and from a number of processors available in a parallel process.

5. (Currently Amended) A computer readable medium storing a program for realizing a parallel processing method for an inverse matrix for a shared memory type scalar parallel computer, comprising:

(a) specifying a square block in a matrix for which an inverse matrix is to be obtained and which is stored in a memory area as an array $A(1:n, 1:n)$ (where n is an order of the matrix) and storing the square block into a subarray $A(\text{nbase}+1:\text{nbase}+\text{iblk}, \text{nbase}, +1:\text{nbase}+\text{iblk})$ (where $\text{nbase}=\text{iblk}*(i-1)$, i is a number of repetition, and iblk is a block width);

(b) decomposing the matrix into upper left, left side, lower left, upper, lower, upper right, right side, and lower right blocks surrounding the square block positioned in the center and storing each divided block into subarrays $A(1:\text{nbase}, 1:\text{nbase})$, $A(\text{nbase}+1:\text{nbase}+\text{iblk}, 1:\text{nbase})$, $A(\text{nbase}+\text{iblk}+1:n, 1:\text{nbase})$, $A(1:\text{nbase}, \text{nbase}+1:\text{nbase}+\text{iblk})$, $A(\text{nbase}+\text{iblk}+1:n, \text{nbase}+1:\text{nbase}+\text{iblk})$, $A(1:\text{nbase},$

nbase+iblk+1:n , $A(\text{nbase+1:nbase+iblk}, \text{nbase+iblk+1:n})$, $A(\text{nbase+iblk+1:n}, \text{nbase+iblk+1:n})$, respectively;

(c) dividing the lower, right side, and lower right blocks of the square block along a longer direction of a row direction or a column direction of each block, assigning each divided block and the square block to each processor and LU decomposing each assigned block by a parallel operation of the processors;

(d) updating the left side, upper, lower, and right side blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor, the parallel operation including execution of a recursive program comprising (1) dividing blocks into a first half and a second half, (2) calculating the first half, (3) applying (1) through (3) to the second half, and further updating in parallel using the blocks updated in the recursive program on the upper left, lower left, upper right, and lower right blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor;

(e) updating the square block in plural stages using one processor by dividing the predetermined square block so that a cost of updating the predetermined square block is within the 1% against a whole cost of calculation; and

(f) setting __nbase of the subarray $A(\text{nbase+1:nbase+iblk}, \text{nbase+1:nbase+iblk})$ to nbase+iblk , and obtaining an inverse matrix of the matrix by repeating the steps of (a) through (f).

6. (Previously Presented) The computer readable medium according to claim 5, wherein

said shared memory type scalar parallel computer comprises a plurality of processors, plural units of cache memory provided for respective processors, plural units of shared memory, and an interconnection network for connection to be made such that the units can be communicated.

7. (Previously Presented) The computer readable medium according to claim 5, wherein

said method is used to realize a Gauss-Jordan method for parallel computation for each block.

8. (Previously Presented) The computer readable medium according to claim 5, wherein

a width of a division used when each block is divided for parallel computation is set such that a total amount of computation of square blocks not processed in parallel can be about 1% of entire computation from a size of a matrix from which an inverse matrix is obtained and from a number of processors available in a parallel process.

9. (New) A shared memory type scalar parallel computer, comprising:

a plurality of processors;

plural units of cache memory provided for respective ones of said processors;

plural units of shared memory that is shared among said processors; and
an interconnection network for connecting said processors, said shared memory units and said cache memory units,

wherein said shared memory type scalar parallel computer is configured and arranged to perform a parallel processing method for an inverse matrix that includes:

(a) specifying a square block in a matrix for which an inverse matrix is to be obtained and which is stored in a memory area as an array $A(1:n, 1:n)$ (where n is an order of the matrix) and storing the square block into a subarray $A(nbase+1:nbase+iblk, nbase+1:nbase+iblk)$ (where $nbase=iblk*(i-1)$, i is a number of repetition, and $iblk$ is a block width);

(b) decomposing the matrix into upper left, left side, lower left, upper, lower, upper right, right side, and lower right blocks surrounding the square block positioned in the center and storing each divided block into subarrays $A(1:nbase, 1:nbase)$, $A(nbase+1:nbase+iblk, 1:nbase)$, $A(nbase+iblk+1:n, 1:nbase)$, $A(1:nbase, nbase+1:nbase+iblk)$, $A(nbase+iblk+1:n, nbase+1:nbase+iblk)$, $A(1:nbase, nbase+iblk+1:n)$, $A(nbase+1:nbase+iblk, nbase+iblk+1:n)$, $A(nbase+iblk+1:n, nbase+iblk+1:n)$, respectively ;

(c) dividing the lower, right side, and lower right blocks of the square block along a longer direction of a row direction or a column direction of each block, assigning each divided block and the square block to each processor and LU decomposing each assigned block by a parallel operation of the processors;

(d) updating the left side, upper, lower, and right side blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor, the parallel operation including execution of a recursive program comprising (1) dividing blocks into a first half and a second half, (2) calculating the first half, (3) applying (1) through (3) to the second half, and further updating in parallel using the blocks updated in the recursive program on the upper left, lower left, upper right, and lower right blocks by a parallel operation of processors including, dividing each block along a longer direction of each block; and assigning divided blocks to each processor;

(e) updating the square block in plural stages using one processor by dividing the predetermined square block so that a cost of updating the predetermined square block is within about 1% against a whole cost of calculation; and

(f) setting $nbase$ of the subarray $A(nbase+1:nbase+iblk, nbase+1:nbase+iblk)$ to $nbase+iblk$, and obtaining an inverse matrix of the matrix by repeating the steps (a) through (f).

10. (New) The shared memory type scalar parallel computer according to claim 9, further comprising realizing a Gauss-Jordan method for parallel computation for each block.

11. (New) The shared memory type scalar parallel computer according to claim 9, wherein a width of a division used when each block is divided for parallel computation is set such that a total amount of computation of square blocks not

processed in parallel can be about 1% of entire computation from a size of a matrix from which an inverse matrix is obtained and from a number of processors available in a parallel process.